GF(2^m)상에서 하이브리드 승산기 및 역원기 설계

강민섭[1)]

Design of Hybrid Multiplier and Inversion Unit Over GF(2^m)

Min-sup Kang[1)]

요 약

타원곡선(ECC) 알고리즘을 구성하는 핵심 연산은 스칼라 곱셈이며, 스칼라 곱셈은 유한체의 곱셈, 덧셈, 그리고 역수 연산으로 이루어져 있다. 본 논문에서는 유한체 GF(2^m) 상에서 연산속도의 고속화를 위해 개선된 구조의 하이브리드(Hybrid) 곱셈기를 제안한다. 제안한 곱셈기는 Bit-Serial 곱셈과 Bit-Parallel 곱셈의 장점을 이용하고 있다. 또한, 효율적인 유한체 역원 연산을 수행하기 위해 기존의 확장 GCD 알고리즘을 기본으로 한 개선된 역원기를 사용한다. 제안된 방법에서 타원곡선은 표준 기저방식(polynomial basis) 으로 표현된다.

핵심어 : 하이브리드 곱셈기, GCD 알고리듬, 역원기, 유한체, Verilog HDL

Abstract

In ECC algorithm, the most time consuming part is scalar multiplication that can be computed by point addition, multiplication and inversion operation. In this paper, we propose a advanced Hybrid multiplier over GF(2^m) for realizing fast computation speed. The proposed multiplier uses a structure combining the advantage of bit-serial multiplication and bit-parallel multiplication. In addition, a modified inversion unit is used for performing an efficient finite field operation based on extended GCD algorithm. In this approach, ECC uses a polynomial basis representation.

Keywords : Hybrid multiplier, GCD algorithm, Inversion unit, Finite field, Verilog HDL

1. 서론

최근 인터넷 기술의 발전과 전자 상거래가 활성화됨에 따라 전자 서명, 인증 및 정보의 암복화 등 보안 기술에 대한 중요성이 대두되고 있다. 정보 보안에 필요한 암호 알고리듬은 키 사용 방법에 따라 공통키(common key encryption) 방식과 공개키(public key encryption) 방법으로 나눌 수 있다. 현재 가장 많이 사용하고 있는 암호 시스템은 RSA(Rivest, Shamir, Adleman)와 타원곡선(ECC: Elliptic Curve Cryptosystem) 이다. ECC는 1985년에 N. Koblitz와 V. Miller에 의해 제안되었고, 이 시스템은 다른 암호시스템에 비해 더 짧은 비트길이의 킷값으로도 안전도가 높은 장점을

접수일(2012년12월15일), 심사의뢰일(2012년12월16일), 심사완료일(1차:2013년01월03일, 2차:2013년01월15일)
게재일(2013년02월28일)

[1]430-714 경기도 안양시 만안구 안양5동, 안양대학교 컴퓨터공학과.
email: mskang@anyang.ac.kr

가진다[1, 2]. ECC는 타원곡선 상에서 유한체 연산이 수행되며, H/W와 S/W로 구현하기가 용이하기 때문에 스마트카드, 무선 통신 단말기 등에 널리 사용되고 있다[2].

타원곡선 알고리즘을 구성하는 핵심 연산은 스칼라 곱셈이며, 스칼라 곱셈은 유한체의 곱셈, 덧셈, 그리고 역수 연산으로 이루어져 있다[1, 2]. 이 곱셈기는 유한체 GF(2^m)상의 임의의 두 원소의 곱셈을 수행하며, 크게 Bit-Serial 곱셈기, Bit-parallel(Cell array)곱셈기, Hybrid 곱셈기로 나눌 수 있다[2-4]. Bit-Serial 곱셈으로 연산을 수행하는 직렬 유한체 곱셈기는 순차회로에 의해서 구성되며, 속도는 느리나, 작은 규모의 하드웨어로 구현 가능한 장점을 가진다[5]. 반면에 Bit-parallel 곱셈기는 m • m개의 cell을 가지는 매우 규칙적인 구조를 가지고 있고 곱셈 결과를 출력하는데 단 한 사이클이 소요된다는 장점을 가지나, m이 커지면 커질수록 하드웨어 오버헤드가 매우 크게 된다[6, 7].

Hybrid 곱셈기는 상기한 두 곱셈기를 혼합한 형태로서 Bit-Serial 곱셈기보다 빠르고, 회로 복잡도는 Bit-parallel 곱셈기보다 매우 낮은 특징을 갖는다. 따라서 유한체상의 곱셈 연산은 타원곡선 암호시스템의 전체적인 성능을 좌우하는 중요한 요소 중의 하나이다.

본 논문에서는 유한체 GF(2^m)상에서 연산속도의 고속화를 위해 개선된 구조의 하이브리드(Hybrid) 곱셈기를 제안한다. 또한, 효율적인 유한체 역원 연산을 수행하기 위해 기존의 확장 유클리드 알고리즘을 수정한 개선된 방법을 제안한다. ECC 암호프로세서는 Xilinx ISE 6.2i 환경에서 Verilog 언어를 이용하여 기술하였고, Xilinx 디바이스 Xc2v8000 을 타겟으로 하여 합성을 수행하였다.

2. 관련 연구

2.1 타원곡선 암호알고리듬

본 논문에서 사용하는 유한체상에서 GF(2^m)상에서 정의되는 비 초특이 타원 곡선 E는 (식 1)과 같다[1-4].

$$y^2 + xy = x^3 + ax^2 + b \quad (a, b \in GF(2^m)) \qquad (식\ 1)$$

타원곡선 E 위의 점들은 점 덧셈 연산에 대해서 군을 이루고, 무한 원점 O는 이 덧셈에 대한 가환군의 항등원이 된다. P(x1, y1)와 Q(x2, y2)를 타원 곡선 E 위의 두 점이라 하면 타원 곡선 상의 덧셈 P + Q = R(x3, y3)는 점 덧셈(Point Addition)과 두배점 덧셈(Point Doubling)을 위한 연산이 필요하다.

① 점 덧셈 즉, P != Q 인 경우 좌표 R은 (식 2)와 같이 구한다.

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$$
$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1$$
$$\lambda = \left(\frac{y_1 + y_2}{x_1 + x_2}\right) \quad \text{(식 2)}$$

② 두배점 덧셈 즉, P == Q 인 경우 좌표 R은 (식 3)와 같이 구한다.

$$x_3 = \lambda^2 + \lambda + a,$$
$$y_3 = x_1^2 + (\lambda + 1)x_3$$
$$\lambda = \left(x_1 + \frac{y_1}{x_1}\right) \quad \text{(식 3)}$$

또한, GF(2^m) 상의 점 역원 연산 알고리즘은 (식 4)과 같이 표현된다.

$$(x3, y3) = -(x1, y1) = (x1, x1+y1) \quad \text{(식 4)}$$

본 논문에서는 타원곡선을 정의하는 파라메터는 SEC2[3] 에서 권장하는 값을 사용하였고, 기약 다항식 f(x)의 차수인 m = 163을 선택하였다. 그리고 f(x)는 polynomial base로 표현되며, $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$ 로 정하였다.

2.2 유한체 산술 연산기 구조

GF(2^m) 타원곡선을 이용한 암호화, 복호화의 핵심 연산은 스칼라 곱셈 연산이다. 스칼라 곱셈 연산은 점 덧셈 연산과 두배점 연산의 연속으로 수행되며, 이들 연산은 유한체 덧셈, 유한체 곱셈, 유한체 나눗셈, 유한체 제곱을 통해 이루어진다[6].

(1) 유한체 GF(2^m) 덧셈기

유한체 GF(2^m)에서의 덧셈은 XOR를 이용해서 간단하게 수행할 수 있다[2]. 또한 GF(2^m)에서는 모든 원소의 덧셈에 대한 역원은 그 자신이므로 GF(2^m)에서의 덧셈과 뺄셈은 동일한 개념이다. GF(2^m) 상의 임의의 두 원소 A와 B를 다항식으로 표현하면 (식 5)와 같다.

$$A = a_0 + a_1\alpha + \cdots + a_{m-1}\alpha^{m-1}$$
$$B = b_0 + b_1\alpha + \cdots + b_{m-1}\alpha^{m-1} \quad \text{(식 5)}$$

그리고 (식 5)에서 두 원소 A와 B의 합을 S라고 하면, S는 (식 6)과 같다.

$$S = A + B$$
$$= (a_0 + b_0) + (a_1 + b_1)\alpha + \cdots + (a_{m-1} + b_{m-1})\alpha^{m-1} \quad \text{(식 6)}$$

(식 6)을 이용하면 유한체 GF(2^m) 덧셈기를 간단히 구현할 수 있다. 이때, 덧셈 연산은 각 비트를 XOR 함으로써 구현된다.

(2) 유한체 GF(2^m) 곱셈기

가. 직렬 유한체 곱셈기

유한체 GF(2^m) 상의 임의의 두 원소를 표준 기저로 표현한 다항식이 각각 (식 7)과 같이 주어진다고 가정하자.

$$A(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_1 x + a_0$$
$$B(x) = b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \ldots + b_1 x + b_0 \quad \text{(식 7)}$$

(식 7)에서 두 원소의 곱은 (식 8)과 같이 정리된다.

$$P(x) = A(x)B(x)$$
$$= A(x)\sum_{i=0}^{m-1} b_i x^i = \sum_{i=0}^{m-1} b_i (x^i A(x)) \quad \text{(식 8)}$$

(식 8)을 풀어서 정리하면 (식 9)와 같은 식을 얻을 수 있다.

$$= b_{m-1}x^{m-1}A(x) + (\ldots + (b_2 x^2 A(x) + b_1 x A(x) + ((b_0 A(x))))\ldots) \quad \text{(식 9)}$$

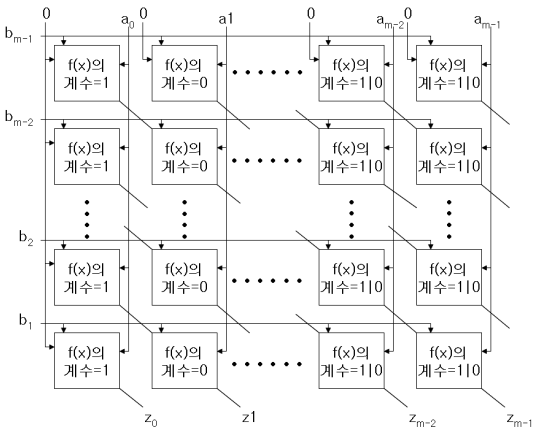
(식 9)를 정리하면, 같은 형태가 반복되는 형태의 (식 10)을 얻을 수 있다.

$$= (\ldots((b_{m-1}A(x)) + b_{m-2}A(x))x + \ldots)x + b_0 A(x) \quad \text{(식 10)}$$

유한체의 계산에서 차수를 감소시키면서 (식 10)의 반복적인 계산을 수행하면, m클럭 사이클만에 유한체 곱셈 결과를 얻을 수 있는 직렬 곱셈기를 구현할 수 있다

나. Bit-parallel 유한체 곱셈기

Serial 곱셈기가 유한체 곱셈 결과를 m사이클 후에 출력한다는 단점을 보완한 곱셈기가 Bit-parallel (Cell array) 곱셈기이다. Cell 구조는 기약다항식의 f(x)의 계수가 0인 경우와 1인 경우의 2개의 형태만 존재하므로 m이 큰 차수의 값을 갖는다고 해도 설계가 용이하다. [그림 1]은 유한체 GF(2^m) 상에서 구현된 Cell array 유한체 곱셈기를 나타낸다.



[그림 1] Bit-parallel 유한체 곱셈기의 구조

[Fig. 1] Structure of bit-parallel GF multiplier

[그림 1]에서 f(x)의 계수가 0인 경우와 1인 경우의 때 Cell을 구현하기 위한 부울 식은 각각 (식 11)과 (식 12)와 같으며, 여기서 i =0, 1, .. m-1을 나타낸다.

$$z\_out = (ai \otimes bi) \oplus z\_in \quad \text{(식 11)}$$

$$z\_out = (ai \otimes bi) \oplus (z\_in \oplus re) \quad \text{(식 12)}$$

이 곱셈기의 복잡도는 $O(m^2)$을 가지게 되므로 m이 커지면 커질수록 하드웨어 오버헤드가 매우 크게 된다.

다. 유한체 GF(2^m) 나눗셈기

유한체 나눗셈 연산에서 가장 먼저 해야 할 일은 제수의 곱셈에 대한 역원을 찾는 일이다. 역원을 구하는 알고리즘으로는 크게 곱셈을 이용한 역원 알고리즘, 유클리드 알고리즘이 사용되고 있다. 곱셈을 이용한 역원 알고리즘은 직교 기저 표현에서는 어떤 원소의 제곱연산이 계수의 비트 circular shift 연산으로 쉽게 구해질 수 있으므로 유용하게 사용될 수 있는 알고리듬이지만, 표준 기저 방식의 표현에서는 효율적이지 못하다[8].

유클리드 알고리즘을 약간 변형한 확장 유클리드 알고리즘(EUA: Extended Euclidean Algorithm)은 유한체 GF(2^m) 상의 임의의 두 원소에 대하여 반복적으로 서로 나누면 최종적으로

최대 공약식을 구하는 방법이다. 이 방법은 표준 연산을 사용하므로 특히 유한체 GF(2^m) 상의 두 원소가 표준 기저 방식으로 표현되었을 때 더 효율적이다. 그림 2는 기존의 유한체 GF(2^m) 상의 나눗셈을 하기 위한 확장 GCD 알고리즘을 나타낸다[8, 9].

```
Input  : G(x), A(x), B(x)
Output : U  has
         P(x) = A(x) / B(x) mod G(x)
Initialize : R = B(x), S = G = G(x),
             U = A(x), V = 0
-----------------------------------
  while S ≠ 0 do
        while r0 == 0 do
            R = R / x
            if u0 == 0 then
                U = U / x;
            else
                U = (U + G) / x;
            end if
        end while
        while s0 == 0 do
            S = S / x
            if v0 == 0 then
                V = V / x;
            else
                V = (V + G) / x;
            end if
        end while
        if S ≥ R then
            (S, R) = (S + R, R);
            (V, U) = (U + V, U);
        else
            (S, R) = (S, S + R);
            (V, U) = (V, U + V);
        end if
    end while
```

[그림 2] 확장 GCD 알고리듬

[Fig. 2] Extended GCD algorithm

[그림 2]에서 A(x), B(x)는 유한체 내에 표준 기저로 표현된 다항식이며, P(x)는 유한체에서 주어진 소수 다항식을 나타낸다.

3. 개선된 유한체 연산기 설계

3.1 Hybrid 유한체 곱셈기의 설계

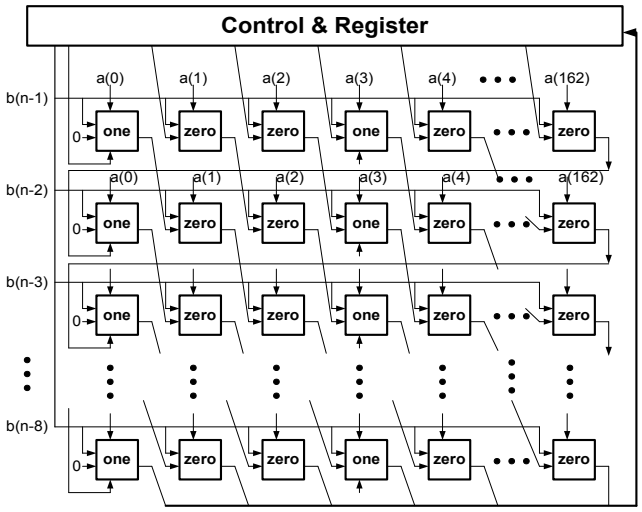
본 논문에서 타원곡선을 정의하는 파라메터는 SEC2[3] 에서 권장하는 값을 사용하며, 기약다항식 f(x)는 차수 m=163을 선택한다. 그리고 f(x)는 표준 기저방식(polynomial basis) 으로 표현되며, $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$ 로 정하였다. 고속연산을 위해서 유한체 곱셈은 직렬 곱셈기와 Cell_array 곱셈기 구조를 혼합한 Hybrid 구조로 설계하였고, 유한체 역원 연산을 하기위해 확장

유클리드 알고리즘을 기본으로 한다. 그리고 이렇게 설계된 유한체 연산기를 사용하여 타원곡선 point 연산기(addition 과 doubling)를 설계하였다. 그리고 point 연산기를 이용해 point multiplication을 구현 하였다.

본 논문에서는 고속 연산을 수행하기 위해 기존의 직렬곱셈기와 Cell array 곱셈기를 혼합한 Hybrid 유한체 곱셈기를 제안한다. GF(2^m) 곱셈식은 (식 13)과 같이 정리할 수 있으며, 곱셈은 MSB에서 LSB 로 수행된다.

$$
\begin{aligned}
Z(a) &= A(a) \cdot B(a) \\
&= A(a) \sum_{i=0}^{m-1} b_i a^i \, Mod(m) \\
&= \sum_{i=0}^{m-1} b_i (a^i A(a)) \, Mod(m) \\
&= (\ldots (A(a) b_{m-1}) a + A(a) b_{m-2}) a \\
&\quad + \ldots) a \, Mod(m) + A(a) b_0 \, Mod(m)
\end{aligned}
\qquad \text{(식 13)}
$$

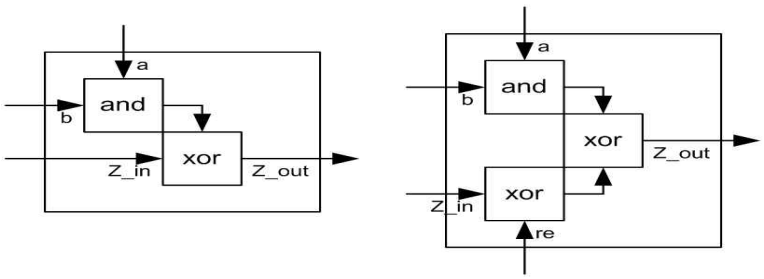
[그림 3]은 (식 13)을 기본으로 하여설계된 차수 n=8인 개선된 Hybrid 곱셈기의 구조를 나타낸다.



[그림 3] 제안된 Hybrid 곱셈기의 구조

[Fig. 3] Proposed Hybrid multiplier

기존의 digit-serial 곱셈기에서는 1bit 씩 순차적으로 계산되던 것과 달리 제안된 방법에서는 m·n 개의 cell 을 가지고 1 clock 에서 n차수의 연산을 수행하게 된다. 기약다항식 $f(x)$의 계수는 0 인 경우와 1 인경우의 2 개의 형태만 존재한다. 본 논문에서는 전자의 경우를 Type-Zero cell, 후자의 경우를 Type-One cell이라 정의한다. [그림 4]는 각각 Type-Zero와 Type-One 셀의 구조를 나타낸다.



(a) Type-Zero 셀 구조 (b) Type-One 셀 구조

[그림 4] 제안하는 셀 구조

[Fig. 4] Proposed cell structure

Type_Zero의 경우는 이전 계산 값에서 reduction 되는 값이 없으므로 Type_One 보다 XOR gate 가 1 개 생략된 구조이다. 제안된 곱셈기를 이용하여 163bit의 유한체 곱셈을 수행할 경우 20 clock과 마지막 소수처리를 위해 1 clock이 필요하게 되므로 총 21 clock에 모든 연산이 완료하게 된다. 따라서 제안된 곱셈기는 종래의 직렬 곱셈기보다 4배정도 처리 속도가 빠르며, 회로의 복잡도는 종래의 Cell array 곱셈기 보다 약 20배 정도 감소된다.

3.2 개선된 확장 유클리드 알고리듬을 이용한 역원기

유한체 산술연산 중에 가장 많은 시간과 복잡한 구조를 필요로 하는 연산기가 바로 나눗셈기이다. 이 알고리즘은 나눗셈 연산에서 역원을 구하는 연산에 추가의 곱셈연산을 하지 않고 역원 연산과 나눗셈 연산에 모두 사용하여 연산속도를 빠르게 할 수 있는 장점을 가진다. 유한체 GF(2^m) 상에서 나눗셈 연산은 유한체 곱셈에서와 마찬가지로 기약다항식에 의존하여 수행되는데, A(x)와 B(x)는 유한체 GF(2^m) 상의 두 원소이고, F(x)는 차수 m의 기약 다항식이고, Z(x)는 B(x)/A(x) mod G(x)의 결과라고 하면, 각각의 다항식은 (식 14)와 같이 표현되며 계수들은 이진수 0 또는 1이다.

$$
\begin{aligned}
A(x) &= a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_1 x + a_0 \\
B(x) &= b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \ldots + b_1 x + b_0 \\
F(x) &= x^m + f_{m-1}x^{m-1} + \ldots + f_1 x + f_0 \\
Z(x) &= z_{m-1}x^{m-1} + z_{m-2}x^{m-2} + \ldots + z_1 x + z_0
\end{aligned}
\qquad \text{(식 14)}
$$

본 논문에서는 유한체 GF(2^m)상의 빠른 나눗셈의 수행을 위해 그림 3의 확장 유클리드 알고리즘을 수정하여 사용한다. 즉, 그림 3의 기존 알고리즘은 전체 while 구문 내에 두 개의 while 구문을 이용하여 확장 유클리드 알고리즘의 일부분인 공약수 구하기를 수행하고 있는데, 위와 같은 방법으로 while 구문을 구성하면 A(x)와 B(x)의 공약수를 구한 후 G(x)와 V(x)의 공약수를 구하게 된

다. 이 알고리듬을 HDL 언어로 구현하였을 경우 하나의 while 구문이 종료된 후에 다른 while 구문이 동작하여 많은 시간을 소모하게 되는 문제점이 있다.

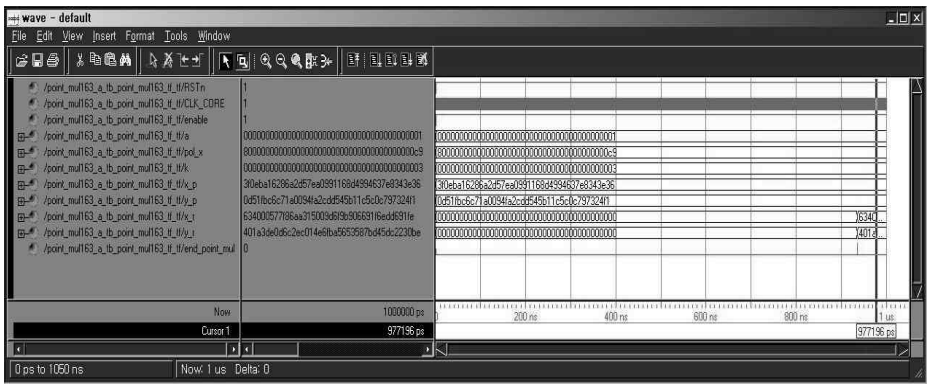
그러나 본 논문에서 제안하는 개선된 알고리즘은 기존의 알고리즘에서 사용하는 while 구문을 개선하여 A(x)와 B(x)의 공약수와 F(x)와 Z(x)의 공약수 구하기를 동시에 병렬적으로 처리 될 수 있도록 구현하였다. 즉, 제안된 알고리즘에서는 전체 while 구문 내에 하나의 while 구문을 구성하고 이 while 구문 내에 두 개의 if 구문으로 수정하여 다음과 같은 하나의 연산식으로 재구성하였다.

```
U = (U + u0 • G) / x;
V = (V + v0 • G) / x;
```

따라서, 이 if 구문은 서로 각기 영향을 주지 않기 때문에 HDL 구문으로 구현시 병행 처리가 가능하기 때문에 고속 연산이 가능하게 된다.

4. FPGA 구현 결과

본 논문에서는 제안된 hybrid 곱셈기와 유한체 역원기를 기본으로 하여 고성능 ECC 프로세서를 구현하였다. 구현된 ECC 프로세서는 Xilinx ISE 6.2i 환경에서 Verilog 언어를 이용하여 기술하였고, Xilinx 디바이스 Xc2v8000 을 타겟으로 하여 합성을 수행하였다. ECC 프로세서의 동작을 검증을 위해 Mentor Graphics사의 ModelSim을 사용하여 시뮬레이션을 수행하였고, 타이밍 시뮬레이션 결과는 [그림 5]와 같다. 여기에서 설계된 ECC 프로세서는 KAIST, IDEC(반도체설계센터)에서 지원받은 툴들을 사용하였다.



[그림 5] 타이밍 시뮬레이션 결과

[Fig. 5] Timing simulation result

시스템의 검증은 [3]에서 구현된 테스트 벡터를 이용하여, 시뮬레이션 결과로부터 제안된 알고리

듬의 정확히 동작함을 확인하였다.

[표 1]은 제안한 유한체 곱셈기에 대한 성능 평가를 나타낸다.

[표 1] 제안된 곱셈기의 성능 평가

[Table 1] Performance evaluation of proposed multiplier

차수(m)	4	8
소요 사이클	41	21
게이트 수	19K	29K
사이클 당 소요시간	9.4ns	12.8ns
총 소요 사이클	386ns	267ns

[표 1]에서 알 수 있듯이 차수를 4와 8로 구현하였을 때의 결과를 보여준다. 차수를 4로 구현하였을 때는 한 사이클당 소요 시간이 약 9 ns 로 짧으며 면적도 적게 사용되지만, 소요 사이클이 41사이클로 전체 연산 수행 시간은 약 386 ns 가 걸린다. 차수를 8차로 구현하였을 때는 사이클당 소요시간과 게이트 수가 증가되지만 전체 연산에 소요되는 사이클이 21사이클로 줄게 되며, 전체 연산 수행 시간은 약 267 ns 로 4차 보다 적은 연산 시간을 갖게 된다.

5. 결론

본 논문에서는 유한체 GF(2^m) 상에서 표준 기저방식(polynomial basis)을 기반으로 한 개선된 하이브리드(Hybrid) 곱셈기 및 역원기 설계를 제안하였다. 제안한 곱셈기는 고속 연산을 수행하기 위해 Bit-Serial 곱셈과 Bit-Parallel 곱셈의 장점을 이용하고 있으며, 역원기는 기존의 확장 유클리드 알고리즘을 개선한 수정된 방법을 사용 하고 있다.

또한, 본 논문에서는 제안된 hybrid 곱셈기와 유한체 역원기를 기본으로하여 고성능 ECC 프로세서를 구현하였다. ECC 암호프로세서는 Xilinx ISE 6.2i 환경에서 Verilog 언어를 이용하여 기술하였고, Xilinx 디바이스 Xc2v8000 을 타겟으로 하여 합성을 수행하였다. ECC 프로세서의 동작을 검증을 위해 Mentor Graphics사의 ModelSim을 사용하여 타이밍 시뮬레이션을 수행하였고, 설계된 시스템이 정확히 동작함을 확인하였다.

참고문헌 [Reference]

[1] Stallings, William, Cryptography and Network Security: Principles and Practice, 2nd Edition. New Jersey: Prentice Hall Inc. (1999)

[2] N.Koblitz, CM-curves with good cryptographic properties, Advances in Cryptology—CRYPTO"91. Lecture Notes in Computer Science, Springer-Verlag, (1992), Vol.576, pp. 279-287.

[3] Certicom Research, SEC2: Recommended Elliptic Curve Cryptography Domain Parameters, (1999)

[4] 6. D. Hankerson, A. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography, Springer-Verlag (2004)

[5] Hero Modares, A Bit-Serial Multiplier Architecture for Finite Fields Over Galois Fields. Journal of Computer Science. (2010) Vol.6, No.11, pp.1237-1246.

[6] A. Reyhani-Masoleh and M. A. Hasan, Low Complexity Bit Parallel Architectures for Polynomial Basis Multiplication over GF(2^m). IEEE Transactions on Computers, (2004) august, Vol.53, No.8, pp.945-959.

[7] F. Rodriguez-Henriquez and C. K. Koc, Parallel Multipliers Based on Special Irreducible Pentanomials. IEEE Transactions on Computers, (2003), Vol.52, No.12, pp.1535-1542.

[8] Hoon Kim, C., Pyo Hong, High-speed division architecture for GF(2^m). Electronics Letters, (2002), Vol.38, pp.835 – 836.

[9] Kaihara, M.E., Takagi, N., A VLSI algorithm for modular multiplication/division. Proceedings of the 16th IEEE Symposium on Computer Arithmetic, (2003) June, pp.220 – 227.

저자 소개

강민섭 (Min-sup Kang)

1984년 한양대학교 전자공학과 졸업 (공학석사)
1992년 (일) 오사카대학교 전자공학과 졸업 (공학박사)
1984년~1992년 한국전자통신연구원 선임연구원
2001년~2002 Univ. of California, Irvine 전기전자공학과 객원연구원
1993년~ 현재: 안양대학교 컴퓨터공학과 교수
관심분야: 정보보호 시스템, 임베디드 시스템, VLSI 설계, RFID/USN